Set Items Description
--- ----- -----------
? t 01304329/7

01304329/7

01304329 SUPPLIER NUMBER: 07501390 (THIS IS THE FULL TEXT)
Using a relational system on Wall Street: the good, the bad, the ugly, and the ideal.
Rozen, Steve; Shasha, Dennis
Communications of the ACM, v32, n8, p988(7)
August, 1989

TEXT:

Using A Relational System On Wall Street: The Good, The Bad, The Ugly, And The Ideal Conventional wisdom has it that complex decision support is an extremely promising application of the relational model. Partly for this reason, one of the authors (Rozen) was hired to design the database for a decision support system used by Wall Street investment managers. The application, based on the Oracle relational database management system, was to replace a previous one written on top of a file system. Problems with the previous application system included the inability to enhance its functionality to suit new analytical applications, the inability to exchange data with other company information, and an antiquated teletype user interface.

The developers were able to reimplement the functionality of the old system in seven months, and their success depended largely on the virtues of the relational model. However, problems still remained. To illustrate both the promise and the problems, in this article, we focus on several critical design requirements and the ability of the relational system to satisfy them. In all important respects, our analysis is independent of Oracle and reflects issues that the use of any relational database management system would raise.

Our view of the relational model is that defined by standard SQL or QUEL. Technically, these are languages equivalent in expressive power to the relational calculus augmented with aggregates (and grouping), a small set of functions on column domains (e.g., addition, substring), a data definition language, update operations, and sorting operators. We assume that relational systems support sharing (concurrency control) and transaction recovery.

THE APPLICATION

The application, called BondDB, is designed to support buying and selling bonds by institutional sales people, traders, and risk managers in an investment bank. Users of BondDB can

(1) try to find investments for a client that are both less expensive than the client's current investments and more profitable as long as interest rates remain below some specific value,

(2) estimate a likely decline in the value of an investment for a typical day, based on the recent variability of a bond's price, to try to keep a trader from holding too many risky bonds, or

(3) evaluate the fair value of an investment today for different future interest rate scenarios.

To perform these analyses, BondDB requires information about the bonds and their historical behavior.

BondDB's financial calculations and user interfaces made it impractical to construct the entire system using only a combination of SQL and application generation tools provided with the database management system. Therefore the application programs are written almost entirely in a conventional programming language.

THE DATA

BondDB contains the following data:

(1) basic information on the characteristics of about 10,000 different bonds,

(2) time series of daily quotes for about 3,000 of these bonds, totaling some 1.8 million points,

(3) about a dozen different kinds of bonds, and

(4) information on portfolios of bonds.

We draw our examples from the representation of two main entity types: bonds and portfolios. Figure 1 is a simplified entity-relationship diagram of BondDB's data-representation requirements.

Bonds

A bond is basically a loan from the holder of the bond to the issuer. It specifies a sequence of payments of principal and interest according to some rules or schedule. Here is a simple example. Issuer: U.S. Treasury Face Value: $1,000.00 Payments: 5/31/88--$38.75 11/30/88--$38.75 5/31/89--$38.75 11/30/89--$38.75 +$1,000.00 face value Type of Bond: Treasury Note Coupon: 7-3/4% Maturity date: 11/30/89

In this case there are four semi-annual interest payments of $38.75, and the principal is paid back in a lump sum after two years.

Portfolios

A portfolio is set of bonds that have been bought in various quantities. Each <bond, quantity> pair is called a position. For some analyses BondDB also needs to record when the bond was bought and for how much. BondDB also includes some information about the portfolio itself, such as the name of its owner, and the order in which to display the portfolio's positions.

BondDB AND THE RELATIONAL MODEL--HOW GOOD A MATCH?

In this section, we discuss central data management requirements of BondDB and how the developers tried to meet them using the relational database management system. We then describe features of an ideal (yet, we believe, realizable) system that could provide superior support.

Management of Bulk Data Types

One of the virtues of the relational model is that statements of its query language operate on entire relations at a time. One can formulate set-oriented expressions such as the following, SELECT BOND_ID FROM BOND WHERE BOND_ID NOT IN (SELECT BOND_ID FROM PRICE); which prints the IDs of all bonds that have no price history (the good).

A bulk data type is a composite data type whose instances contain a dynamically varying, and potentially large, number of elements. Thus sets and relations are bulk data types, as are multisets, sequences (lists), trees, and graphs. BondDB required usable implementations of bulk data types other than relations and the ability to include instances of such types as attributes of bonds and portfolios.

For example, many bonds are characterized by a sequence of time intervals and prices indicating when and for how much the issuer can call the bond (i.e., prepay the loan to the bond holder). An example is the following: Issuer: Brooklyn Union Gas Type of Bond: Preferred stock Last dividend: $ 2.47 Face value: $25.00 Moody's Rating: A2 SP rating: A 'Call' schedule: 8/31/81 to 8/30/86--$27.92 8/31/86 to 8/30/91--$27.36 8/31/91 to 8/30/96--$26.81 8/31/96 to 1/01/15--$26.25

In this case, the issuer can call the bond at $27.36 per share any time between 8/31/86 and 8/30/91. The most common query is to retrieve the call information when the other information about this security is fetched. A more sophisticated query might want to know the monetary value of the calls, which is a function of the bond's current price, price volatility, time to maturity, and remaining call schedule. Both queries require manipulation of the CALL SCHEDULE as a sequence.

Representing the call schedule with multiple attributes, that is, BOND(. . ., CALL_DATE.sub.1, CALL_PRICE.sub.1, . . ., CALL_DATE.sub.1, CALL_PRICE.sub.1, . . .), would lead to well-known problems.

One of the problems is that maximum size of a call schedule may not be known when the schema is designed. If CALL.sub.n+1 is added, all code using call schedules will need to be modified. Furthermore, variables in SQL cannot range over columns. For example, to print just the remaining call dates and prices, we would have to write: SELECT CALL_DATE.sub.1, CALL_PRICE.sub.1 FROM BOND WHERE CALL_DATE.sub.1 > TODAY; . . . SELECT CALL_DATE.sub.n, CALL_PRICE.sub.n FROM BOND WHERE CALL_DATE.sub.n > TODAY;

Therefore the developers include a separate table for this information for all bonds in the database: CALL(BOND_ID, CALL_DATE, CALL_PRICE), where BOND_ID and CALL_DATE constitute the key. The information for the Brooklyn Union Gas (BUG) example is then stored in two tables, BOND and CALL as seen Figure 2. To fetch all information about the bond BUG-PS from a relational system into a host language one needs two queries (the ugly): SELECT* FROM BOND WHERE BOND_ID = 'BUG-PS'; and SELECT* FROM CALL WHERE BOND_ID = 'BUG-PS' ORDER BY CALL_DATE;

Ideally, one would like to express this relationship directly by defining an attribute CALL_SCHEDULE of type SEQUENCE (of tuples) on the BOND table: CREATE TABLE BOND (BOND_ID CHAR(10), . . . CALL_SCHEDULE

SEQUENCE (CALL_DATE DATE, CALL_PRICE FLOAT,) . . .);

Then to retrieve this information one would simply write: SELECT* FROM BOND;

This would be followed by operations that can handle sequences as standard relational query languages handle relations.

Procedures as Data

BondDB's developers tood advantage of the capability of relational systems to store queries as views. For example, the following statements basically store a procedure to ensure that only the owner of a portfolio can modify it. CREATE VIEW UPDATE_PORTFOLIO AS SELECT* FROM ALL_PORTFOLIO WHERE ALL_PORTFOLIO.OWNER = USER; GRANT SELECT, UPDATE, INSERT, DELETE ON UPDATE_PORTFOLIO TO PUBLIC;

In this example the stored equity, which takes advantage of the set-at-time relational operators, provides a succinct, executable specification of the modifiability constraints on portfolios (the good).

Some relational systems offer triggers, i.e., statements to be executed whenever some change to the data, such as an insert, occurs. Many database management system implementors consider triggers to be an exotic feature, but they could have been useful to BondDB's developers. For example, in BondDB, to make the most recent price quickly available, it is stored with the bond characteristic information in the BOND table. Thus an insert or update of the PRICE table may actually require modifications to two tables, PRICE and BOND. BondDB could have used triggers to ensure that the prices stored in the BOND table were always up-to-date as shown in Figure 3. For a DBMS with triggers, application programs would need to modify PRICE only. If the need to maintain the most recent price disappeared, or if it became necessary to maintain the most recent five prices, there would be no need to modify the application programs, because the update action on PRICE would be encapsulated in the database.

BondDB's need to store procedures went further than triggers and views. Some data can be stored succinctly in the form of a rle, but viewed by client software as the data generated by the rule (an extension of the view concept). For example, in some, but not all, call schedules, a sequence of rows can be expressed as a rule, e.g., BUG-PS becomes callable on 8/31/81 at $27.92; thereafter the call price declines three times by 55-2/3 cents at five-year intervals; the call price is rounded to the nearest cent.

Other information can be expressed only procedurally, such as rules that determine the coupon of floating rate bonds based on the London interbank lending rate and the treasury bill rate. For example, the following formula determines one bond's coupon payments:

A typical query is: For a given future interest rate scenario, and during a given time period, what payments will the holder receive?

In cases such as these it would be best if the database could store the procedure but allow client software to view the data as a sequence (the ideal). What the developers have to do now is to represent BUG-PS as the sequence that host-language applications must manipulate (the ugly). They do not represent the lending-rate-dependent data at all, since it depends on values that are unknown in advance (the bad).

Data Abstraction

Initially, BondDB's developers represented financial entities by a straightforward transcription of the normalized relational schema into C structs. The developers assumed it would be satisfactory to supply application programmers with routines which return query results as arrays of structs. They also assumed that the programmers could be relied on to manipulate the struct representations of database tuples appropriately. There were two main flaws with the approach:

(1) different application programmers tended to write routines with overlapping functionality, and

(2) some kinds of bonds have important information in more than one table (e.g., the BOND and CALL tables in Figure 2) because of normalization, making it difficult for the progammer to know which tables contain information about a given bond.

Thus this approach caused reliability problems from the beginning.

Eventually, the developers realized that bonds and portfolios are modified and analyzed in certain stereo-typical ways. For example, portfolios may be fetched from or written to the database, and manipulated or analyzed using mathematical tools. So, to an application, the portfolio is a collection of data with some set of operations that can be issued on that data. This is the definition of an abstract data type.

When the developers recognized the abstract data type character of entities represented in BondDB, they defined ADTs in C to represent

BondDB's major financial constructs, (e.g., bonds and portfolios). The developers use the relational system for its virtues of ensuring recovery and concurrency control, associative access, the facility with which they can set up and modify relational schemas, and the ease with which they can generate query code for tables. In order to operate on an entity, the developers must map the date from relations to the C-language structures of the ADT, manipulate it using C-based abstract data type primitives, and then possibly return it to relations.

Ideally, the developers would define the abstract data types directly in the database system. Putting ADTs into the database dovetails well with the ideals of the previous sections, because

(1) BondDB's ADTs have component instances of bulk data types,

(2) one would like to store the operations of the ADT in the database as well as the data, and

(3) sufficiently powerful ADT facilities could be used to define new bulk data types if necessary.

Physical Data Independence

The relational model provides the application program with independence from internal data structures in three main ways:

(1) Queries will have the same result regardless of the physical organization of the tables and the availability or unavailability of indexes.

(2) Application programmers need to be concerned with the order of columns in tables or with columns that are not referenced since columns are referenced by name rather than by offset from the beginning of a record, for example.

(3) Stored procedures in the form of views let the database management system present virtual tables, providing an optional additional layer of independence.

BondDB developers relied on all three kinds of physical data independence (the good).

The options for specifying details of physical implementation, however, did not always provide enough efficiency for BondDB (the bad). Performance considerations forced the developers to implement part of the database using operating system facilities. In particular, they installed a copy of the BOND table in shared memory, thus partly duplicating the function of the database management system's cache (the ugly). At that point they had to hand-code search structures since a linear search of the shared memory consumed more CPU resources than a lookup in the database using indexes.

The developers also were forced to rely on reserved fields to enable them to incorporate additional columns to the C struct without simultaneously re-compiling and re-linking all application programs. The management headaches introduced in maintaining the copied data were substantial. For example, whenever important data needed to be corrected, the shared memory copy had to be reinitialized making it unavailable for about 15 minutes. Furthermore, there was a period of several hours each day when the shared memory copy did not agree with the database copy.

To say that, ideally, the hardware should have run faster is not helpful. At the least, an ideal database management system should provide more options for specifying implementation aspects. In this example, being able to fix a table in memory would have been helpful.

Recovery and Concurrency Control

There is one final issue we observed that ties the physical level to the abstract level. One of the important steps forward in database systems was the decoupling of concurrency control and recovery from the data model (relational, network, or hierarchical). This decoupling is not always desirable, at least for BondDB.

Users must be able to modify portfolios when they buy or sell bonds in them. Each modification may require several operations. For example, to add a bond, a user provides the characteristics of the bond (if it does not already exist in the database) and the quantity bought. To make this set of operations atomic (i.e., all-or-nothing) in a file system which only provides atomic operations on records, the developers would have to order the operations carefully and provide adequate redundancy to ensure recoverability. For example, when adding a position with a bond not yet in the database, one would first save the bond characteristic information, and then the position because without the bond characteristic information, the position is meaningless.

The transaction notion, supported by relational systems, is a big improvement over the facilities usually provided by a file system (the good). Transaction support automatically provides recoverability at the

level of a whole group of operations constituting, for example, the addition of a bond to a portfolio.

Implementing such transactions normally requires holding update locks until the end of the transaction (to prevent other transactions from reading uncommitted updates). In most systems the smallest item that can be locked is a page, and BondDB stored several portfolio positions per page. Unfortunately, there is a small set of actively modified portfolios to which most position records are added. This results in records from different portfolios interleaved on the same pages being updated concurrently. The developers therefore frequently received the complaint that a user was locked out of his portfolio when no one else was accessing it (the bad).

To overcome this problem, the developers defined a protocol in which processes only select for update (i.e., exclusively lock) the header record (the one containing the portfolio ID, the owner, and so on). All other portfolio information (hundreds of positions) is selected from the database without locking. BondDB then modifies the portfolio in user space. This works, but is incovenient because to avoid unnecessary database updates, BondDB must keep track of what data in the portfolio has been modified and what has not. Code to keep track of how the portfolio has changed is hard to debug and test. Furthermore, there is the danger that an interactive SQL user, unaware of the protocol, might accidentally circumvent it. In an ideal system the unit of locking would be a portfolio, which is an abstract data type, so two users could modify different portfolios concurrently.

RELATED WORK

Codd has faulted the SQL language for inadequate handling of null values and other design problems, and has faulted SQL implementations for permitting duplicate rows. Here we assess the utility of the relational model per se, rather than a particular query language. For example, some systems may permit duplicate rows partly out of real need for multi-relations. Associated difficulties are often due to inconsistent or obscure rules for handling the duplicate rows.

Much criticism of the relational model focuses not on its possible shortcomings for traditional business applications, but on either

(1) its inadequate expressiveness for non-traditional database applications, such as design databases, cartographic databases, and multi-media databases, or

(2) its inadequate performance for high transaction applications, notwithstanding recent advances in high-transaction systems such as NonStop SQL.

Sometimes both criticisms play a role, as in Peinl, Reuter, and Sammer's discussion of a bid-matching system for a stock exchange. In this system problems arise because the relational system has inadequate knowledge of the semantics of the ADTs represented in it and therefore cannot make optimizations necessary for the application's stringent performance requirements.

Nevertheless, some researchers have observed the shortcomings that BondDB encountered. For example, Bancilhon and Maier assert that current systems are not satisfactory for business applications because of the impedance mismatch between the relational system and the host language and then go on to discuss possible improvements. The impedance mismatch is the discontinuity in type system and computational model between the database management system and the host language; the database management system utilizes set-at-a-time expressions with a non-procedural language, while the host language offers record- and field-level expressions in an imperative language. Indeed the impedance mismatch was one of the factors that forced BondDB's developers to manage two representations of each financial entity and supply translation code between the two.

Many workers are constructing systems that might be more satisfactory. The design space is large, but the work is promising.

Stonebraker, citing the success of the relational model, proposes evolutionary changes in POSTGRES, which supports user-defined data types and complex objects via the storage of procedures as attribute values. POSTGRES also allows lower-level access when performance requirements demand it. Another approach to extending the relational model, by allowing nested relations, is taken by AIM-II and R.sup.2.D.sup.2., which supports, in addition, user defined ADTs for complex objects.

Many projects (e.g., ENCORE, GARDEN, Iris, O.sub.2., ORION, PROTEUS, Trellis/Owl) and even products (e.g., OPAL/Gemstone, PROBE, Vbase) have begun to build object-oriented database management systems whose basic persistent objects are ADT instances.

Some of the systems above, and others (e.g., Starburst) allow the

application developer to include additional database access structures. Other systems (e.g., EXODUS, GENESIS) provide a set of building blocks from which to assemble a database management system tailored to a particular application. Finally, some researchers are attempting to erase the distinction between the programming language and the database management system altogether. Examples of this approach are Galileo, and PS-Algol.

CONCLUSION

This article attempted an empirical assessment of the tools BondDB's developers had at their disposal--a relational database management system with a conventional programming language.

We saw that, while relational systems are good, their limitations require ugly workarounds, even though realizable improvements would result in a cleaner application implementation.

This was by no means a complete listing of all issues the developers faced. Some, such as politics, are pervasive but immune to a technological solution. Others, like the need for rapid development in a competitive environment depend on factors other than the data model such as the availability of people and machine resources. Application programmers do not ask for magic, however, just the best tools that can be efficiently implemented.

Necessary features of an ideal system begin to emerge. It would:

(1) include a richer selection of built-in bulk data types than relations only, and support data abstraction to allow users to introduce instances of their own complex data types--our example was call schedule sequences and corresponding operations as database values,

(2) allow procedures written in arbitrary programming languages to be stored in the database and viewed either as procedures or as the data that they produce at any given time--our example was floating rate coupons,

(3) offer more tuning options, such as the ability to fix certain data in main memory, and

(4) allow locking of logical objects, such as portfolio instances.

To summarize our challenge to designers of future database management systems: Go ahead, make our database.

Acknowledgments. This work was partially supported by the Office of Naval Research under grant N00014-85-K-0046. Special thanks to Norman M. Birkett, David M. Siegel, and Gudmundur Vigfusson, who helped define BondDB's information architecture, for many enlightening discussions on what would be better for BondDB, and to Yosi Ben-Dov for balancing the short-term exigencies of the users against the long-term exigencies of maintainable development. We also thank Norman M. Birkett, Gudmundur Vigfusson, and the anonymous reviewers for suggesting improvements.

?

```
? show files;ds
File 348:EUROPEAN PATENTS 1978-2006/ 200616
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20060420,UT=20060413
         (c) 2006 WIPO/Univentio

Set     Items   Description
S1     246847   ASSET? ? OR STOCK OR BONDS OR SECURITIES OR FUNDS OR FINAN-
              CIAL OR INVESTMENT? ? OR PORTFOLIO? ? OR RETIREMENT()ACCOUNT?
              ? OR 401K OR 401()K
S2      14299   S1(6N)(ABSTRACT? OR PARSE? OR PARSING OR SEGREGAT? OR SEPA-
              RAT? OR DIVID? OR DIVISION? OR CLASSIFY? INDEX? OR SPLIT? OR -
              SEGMENT? OR APPORTION? OR PORTION? OR SECTION?)
S3        436   S2(6N)(REPRESENTATION()(FRAMEWORK OR FORMAT? ?) OR MULTILE-
              VEL? ? OR MULT()LEVEL? ? OR SUBGROUP? ? OR SUB()GROUP? ? OR C-
              ATEGORIZ? OR CATEGORIS? OR CLASS OR CLASSES OR SUBJECTS OR AR-
              EAS OR TYPE)
S4     681720   RULE? ? OR GUIDELINE? ? OR POLICY OR POLICIES OR PROCEDURE?
               ? OR DICTIONARY OR META()DATA OR META()DATA OR DATA(1N)(DESC-
              RIPTION OR TYPE? ?)
S5        317   S4(8N)ABSTRACTION
S6      41104   HIERARCH?
S7      13275   XML OR "C++"
S8          1   S2(10N)S5
S9        239   S2(10N)S4
S10         8   S3(10N)S9
S11         0   S3(10N)S5
S12         0   S3(30N)S5
S13        16   S3 AND S5
S14         0   S10(10N)(STORAGE OR SERVER? OR DATA()WAREHOUSE OR DATABASE)
S15         0   (S6 OR S7)(10N)S10
S16         6   (S6 OR S7) AND S10
S17        23   S8 OR S13 OR S16
S18        25   S8 OR S10 OR S13 OR S16 OR S17
? t18/3,k/all

 18/3,K/1     (Item 1 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS
(c) 2006 European Patent Office. All rts. reserv.

01250736
Data processing method and system utilizing parallel processing
Datenverarbeitungsverfahren und paralleles Verarbeitungssystem
Methode de traitement de donnees et systeme de traitement parallele
PATENT ASSIGNEE:
  Mouradian, Gary C., (2835200), 40481 North Sunset Drive, Antioch, IL
    60002, (US), (Applicant designated States: all)
INVENTOR:
  Mouradian, Gary C., 40481 North Sunset Drive, Antioch, IL 60002, (US)
LEGAL REPRESENTATIVE:
  Quintelier, Claude et al (73881), Gevers & Vander Haeghen, Patent
    Attorneys, Rue de Livourne 7, 1060 Brussels, (BE)
PATENT (CC, No, Kind, Date):  EP 1079309  A1  010228 (Basic)
APPLICATION (CC, No, Date):   EP 99870176 990820;
DESIGNATED STATES: DE; FR; GB; IT; NL
EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO; SI
INTERNATIONAL PATENT CLASS (V7): G06F-015/18; G06N-003/00
ABSTRACT WORD COUNT: 190
NOTE:
  Figure number on first page: 4
LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text  Language   Update    Word Count
      CLAIMS A  (English)  200109      900
      SPEC A    (English)  200109    31208
Total word count - document A         32108
Total word count - document B             0
Total word count - documents A + B    32108

...SPECIFICATION into entirely new kinds of functionalities and performance
  capabilities. In Patent 5,305,393, 'Exhaustive  Hierarchical  Near
  Neighbor Operations On An Image", Mahoney offers a parallel processing
```

```
? b 411; sf all

       21apr06 11:45:57 User249839 Session D7595.1
            $0.00    0.213 DialUnits FileHomeBase
     $0.00  Estimated cost FileHomeBase
     $0.03  TELNET
     $0.03  Estimated cost this search
     $0.03  Estimated total session cost   0.213 DialUnits

File 411:DIALINDEX(R)

DIALINDEX(R)
   (c) 2006 Dialog

*** DIALINDEX search results display in an abbreviated ***
*** format unless you enter the SET DETAIL ON command. ***
   You have 562 files in your file list.
   (To see banners, use SHOW FILES command)
? s (abstraction()rule? ?)(10n)(asset? ? or stock or bonds or securities or funds or
financial or investment? ? or portfolio? ? or retirement()account? ? or 401k or 401()
k) not py>2000

Your SELECT statement is:
   s (abstraction()rule? ?)(10n)(asset? ? or stock or bonds or securities
or funds or financial or investment? ? or portfolio? ? or
retirement()account? ? or 401k or 401()k) not py>2000

           Items   File
           -----   ----
       Examined  50 files
       Examined 100 files
       Examined 150 files
       Examined 200 files
       Examined 250 files
       Examined 300 files
       Examined 350 files
       Examined 400 files
       Examined 450 files
       Examined 500 files
       Examined 550 files
Processing

   No files have one or more items; file list includes 562 files.
   One or more terms were invalid in 106 files.

?
```

```
? show files;ds
File 625:American Banker Publications 1981-2006/Apr 21
         (c) 2006 American Banker
File 268:Banking Info Source 1981-2006/Apr W3
         (c) 2006 ProQuest Info&Learning
File 626:Bond Buyer Full Text 1981-2006/Apr 21
         (c) 2006 Bond Buyer
File 267:Finance & Banking Newsletters 2006/Apr 17
         (c) 2006 Dialog
File 139:EconLit 1969-2006/Apr
         (c) 2006 American Economic Association

Set     Items   Description
S1     996163   ASSET? ? OR STOCK OR BONDS OR SECURITIES OR FUNDS OR FINAN-
             CIAL OR INVESTMENT? ? OR PORTFOLIO? ? OR RETIREMENT()ACCOUNT?
             ? OR 401K OR 401()K
S2      60293   S1(6N)(ABSTRACT? OR PARSE? OR PARSING OR SEGREGAT? OR SEPA-
             RAT? OR DIVID? OR DIVISION? OR CLASSIFY? INDEX? OR SPLIT? OR -
             SEGMENT? OR APPORTION? OR PORTION? OR SECTION?)
S3        875   S2(6N)(REPRESENTATION()(FRAMEWORK OR FORMAT? ?) OR MULTILE-
             VEL? ? OR MULT()LEVEL? ? OR SUBGROUP? ? OR SUB()GROUP? ? OR C-
             ATEGORIZ? OR CATEGORIS? OR CLASS OR CLASSES OR SUBJECTS OR AR-
             EAS OR TYPE)
S4     448258   RULE? ? OR GUIDELINE? ? OR POLICY OR POLICIES OR PROCEDURE?
              ? OR DICTIONARY OR META()DATA OR META()DATA OR DATA(1N)(DESC-
             RIPTION OR TYPE? ?)
S5          8   S4(8N)ABSTRACTION
S6       6717   HIERARCH?
S7       1131   XML OR "C++"
S8          1   S2(10N)S5
S9       1855   S2(10N)S4
S10        23   S3(10N)S9
S11         0   S3(10N)S5
S12         0   S3(30N)S5
S13         0   S3 AND S5
S14         0   S10(10N)(STORAGE OR SERVER? OR DATA()WAREHOUSE OR DATABASE)
S15         0   (S6 OR S7)(10N)S10
S16         3   (S6 OR S7) AND S10
S17         4   S8 OR S13 OR S16
S18         3   RD  (unique items)
S19         4   S1 AND S5
S20         2   S19 NOT PY>2000
S21         2   RD  (unique items)
S22        31   S5 OR S8 OR S10 OR S16:S21
S23        30   RD  (unique items)
? t23/3,k/all

 23/3,K/1     (Item 1 from file: 625)
DIALOG(R)File 625:American Banker Publications
(c) 2006 American Banker. All rts. reserv.

0256511
* Further Easing Sought In Merchant Bank Regs
American Banker - April 3, 2001; Pg. 1; Vol. 166, No. 64
DOCUMENT TYPE: Journal     LANGUAGE: English     RECORD TYPE: Fulltext
WORD COUNT:       921

BYLINE:
BY ROB GARVER

TEXT:
...take such investments into
account.
    "We still don't understand the need for separate capital rules for
merchant banking," he said. "This is the only asset class separated
 out for
special treatment."
    Others scheduled to testify are John P. Whaley, a partner with...


 23/3,K/2     (Item 2 from file: 625)
DIALOG(R)File 625:American Banker Publications
```

```
? show files;ds
File  15:ABI/Inform(R) 1971-2006/Apr 21
         (c) 2006 ProQuest Info&Learning
File  16:Gale Group PROMT(R) 1990-2006/Apr 21
         (c) 2006 The Gale Group
File 148:Gale Group Trade & Industry DB 1976-2006/Apr 21
         (c)2006 The Gale Group
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 275:Gale Group Computer DB(TM) 1983-2006/Apr 20
         (c) 2006 The Gale Group
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Apr 21
         (c) 2006 The Gale Group
File   9:Business & Industry(R) Jul/1994-2006/Apr 20
         (c) 2006  The Gale Group
File  20:Dialog Global Reporter 1997-2006/Apr 21
         (c) 2006 Dialog
File 476:Financial Times Fulltext 1982-2006/Apr 22
         (c) 2006 Financial Times Ltd
File 610:Business Wire 1999-2006/Apr 21
         (c) 2006 Business Wire.
File 613:PR Newswire 1999-2006/Apr 21
         (c) 2006 PR Newswire Association Inc
File  24:CSA Life Sciences Abstracts 1966-2006/Mar
         (c) 2006 CSA.
File 634:San Jose Mercury  Jun 1985-2006/Apr 20
         (c) 2006 San Jose Mercury News
File 636:Gale Group Newsletter DB(TM) 1987-2006/Apr 20
         (c) 2006 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File  13:BAMP 2006/Apr W2
         (c) 2006  The Gale Group
File  75:TGG Management Contents(R) 86-2006/Apr W2
         (c) 2006 The Gale Group
File  95:TEME-Technology & Management 1989-2006/Apr W3
         (c) 2006 FIZ TECHNIK
File 348:EUROPEAN PATENTS 1978-2006/ 200616
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20060420,UT=20060413
         (c) 2006 WIPO/Univentio
File 350:Derwent WPIX 1963-2006/UD,UM &UP=200625
         (c) 2006  Thomson Derwent
File 344:Chinese Patents Abs Jan 1985-2006/Jan
         (c) 2006 European Patent Office
File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
         (c) 2006 JPO & JAPIO
File 371:French Patents 1961-2002/BOPI 200209
         (c) 2002 INPI.  All rts. reserv.
File   2:INSPEC 1898-2006/Apr W2
         (c) 2006 Institution of Electrical Engineers
File  35:Dissertation Abs Online 1861-2006/Mar
         (c) 2006 ProQuest Info&Learning
File  65:Inside Conferences 1993-2006/Apr 21
         (c) 2006 BLDSC all rts. reserv.
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/Mar
         (c) 2006 The HW Wilson Co.
File 256:TecInfoSource 82-2006/May
         (c) 2006 Info.Sources Inc
File 474:New York Times Abs 1969-2006/Apr 20
         (c) 2006 The New York Times
File 475:Wall Street Journal Abs 1973-2006/Apr 20
         (c) 2006 The New York Times
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13
         (c) 2002 The Gale Group
File  23:CSA Technology Research Database 1963-2006/Apr
         (c) 2006 CSA.
File  56:Computer and Information Systems Abstracts 1966-2006/Apr
         (c) 2006 CSA.
File  94:JICST-EPlus 1985-2006/Jan W4
```

```
? show files;ds
File  15:ABI/Inform(R) 1971-2006/Apr 21
         (c) 2006 ProQuest Info&Learning
File  16:Gale Group PROMT(R) 1990-2006/Apr 21
         (c) 2006 The Gale Group
File 148:Gale Group Trade & Industry DB 1976-2006/Apr 21
         (c)2006 The Gale Group
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 275:Gale Group Computer DB(TM) 1983-2006/Apr 20
         (c) 2006 The Gale Group
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Apr 21
         (c) 2006 The Gale Group

Set     Items   Description
S1   13419583   ASSET? ? OR STOCK OR BONDS OR SECURITIES OR FUNDS OR FINAN-
             CIAL OR INVESTMENT? ? OR PORTFOLIO? ? OR RETIREMENT()ACCOUNT?
             ? OR 401K OR 401()K
S2    1127303   S1(6N)(ABSTRACT? OR PARSE? OR PARSING OR SEGREGAT? OR SEPA-
             RAT? OR DIVID? OR DIVISION? OR CLASSIFY? INDEX? OR SPLIT? OR -
             SEGMENT? OR APPORTION? OR PORTION? OR SECTION?)
S3      20199   S2(6N)(REPRESENTATION()(FRAMEWORK OR FORMAT? ?) OR MULTILE-
             VEL? ? OR MULT()LEVEL? ? OR SUBGROUP? ? OR SUB()GROUP? ? OR C-
             ATEGORIZ? OR CATEGORIS? OR CLASS OR CLASSES OR SUBJECTS OR AR-
             EAS OR TYPE)
S4    5349171   RULE? ? OR GUIDELINE? ? OR POLICY OR POLICIES OR PROCEDURE?
              ? OR DICTIONARY OR META()DATA OR META()DATA OR DATA(1N)(DESC-
             RIPTION OR TYPE? ?)
S5        459   S4(8N)ABSTRACTION
S6     136425   HIERARCH?
S7     137949   XML OR "C++"
S8          4   S2(10N)S5
S9      18365   S2(10N)S4
S10       248   S3(10N)S9
S11         0   S3(10N)S5
S12         0   S3(30N)S5
S13         1   S3 AND S5
S14         0   S10(10N)(STORAGE OR SERVER? OR DATA()WAREHOUSE OR DATABASE)
S15         0   (S6 OR S7)(10N)S10
S16         4   (S6 OR S7) AND S10
S17         8   S8 OR S13 OR S16
S18         7   RD  (unique items)
? t18/3,k/all

 18/3,K/1     (Item 1 from file: 15)
DIALOG(R)File  15:ABI/Inform(R)
(c) 2006 ProQuest Info&Learning. All rts. reserv.

02996587  954864281
Ageing  and  Pension  System Reform: IMPLICATIONS FOR FINANCIAL MARKETS AND
ECONOMIC POLICIES
Visco, Ignazio
Financial Market Trends  PP: 1-3, 9-111, 113-114  Nov 2005
ISSN: 0378-651X  JRNL CODE: FMT
WORD COUNT: 26937

...TEXT: supervisory oversight. In this respect it may be useful to pursue
further consistency between some  areas  of regulation of pension  funds
and insurance companies. This  section  discusses key  policy  issues that
may be considered in national reform projects in order to provide the right
...Brooks, 1998, Chinn and Prasad, 2003, and Luhrmann 2003). Overall, this
literature suggests a clear  hierarchy . Young countries should experience
current account deficits, as investment demand outstrips domestic saving.
So should...


 18/3,K/2     (Item 2 from file: 15)
DIALOG(R)File  15:ABI/Inform(R)
(c) 2006 ProQuest Info&Learning. All rts. reserv.

02305947  102902757
Structuring and managing a community bank loan review program
```

```
? show files;ds
File 350:Derwent WPIX 1963-2006/UD,UM &UP=200625
         (c) 2006  Thomson Derwent
File 344:Chinese Patents Abs Jan 1985-2006/Jan
         (c) 2006 European Patent Office
File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
         (c) 2006 JPO & JAPIO
File 371:French Patents 1961-2002/BOPI 200209
         (c) 2002 INPI.  All rts. reserv.
File   2:INSPEC 1898-2006/Apr W2
         (c) 2006 Institution of Electrical Engineers
File  35:Dissertation Abs Online 1861-2006/Mar
         (c) 2006 ProQuest Info&Learning
File  65:Inside Conferences 1993-2006/Apr 21
         (c) 2006 BLDSC all rts. reserv.
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/Mar
         (c) 2006 The HW Wilson Co.
File 256:TecInfoSource 82-2006/May
         (c) 2006 Info.Sources Inc
File 474:New York Times Abs 1969-2006/Apr 20
         (c) 2006 The New York Times
File 475:Wall Street Journal Abs 1973-2006/Apr 20
         (c) 2006 The New York Times
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13
         (c) 2002 The Gale Group
File  23:CSA Technology Research Database 1963-2006/Apr
         (c) 2006 CSA.
File  56:Computer and Information Systems Abstracts 1966-2006/Apr
         (c) 2006 CSA.
File  94:JICST-EPlus 1985-2006/Jan W4
         (c)2006 Japan Science and Tech Corp(JST)
>>>No sets currently exist
?
```

```
? show files;ds
File   9:Business & Industry(R) Jul/1994-2006/Apr 20
         (c) 2006  The Gale Group
File  20:Dialog Global Reporter 1997-2006/Apr 21
         (c) 2006 Dialog
File 476:Financial Times Fulltext 1982-2006/Apr 22
         (c) 2006 Financial Times Ltd
File 610:Business Wire 1999-2006/Apr 21
         (c) 2006 Business Wire.
File 613:PR Newswire 1999-2006/Apr 21
         (c) 2006 PR Newswire Association Inc
File  24:CSA Life Sciences Abstracts 1966-2006/Mar
         (c) 2006 CSA.
File 634:San Jose Mercury  Jun 1985-2006/Apr 20
         (c) 2006 San Jose Mercury News
File 636:Gale Group Newsletter DB(TM) 1987-2006/Apr 20
         (c) 2006 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc

Set     Items   Description
S1   18868539   ASSET? ? OR STOCK OR BONDS OR SECURITIES OR FUNDS OR FINAN-
             CIAL OR INVESTMENT? ? OR PORTFOLIO? ? OR RETIREMENT()ACCOUNT?
             ? OR 401K OR 401()K
S2    1281605   S1(6N)(ABSTRACT? OR PARSE? OR PARSING OR SEGREGAT? OR SEPA-
             RAT? OR DIVID? OR DIVISION? OR CLASSIFY? INDEX? OR SPLIT? OR -
             SEGMENT? OR APPORTION? OR PORTION? OR SECTION?)
S3      45779   S2(6N)(REPRESENTATION()(FRAMEWORK OR FORMAT? ?) OR MULTILE-
             VEL? ? OR MULT()LEVEL? ? OR SUBGROUP? ? OR SUB()GROUP? ? OR C-
             ATEGORIZ? OR CATEGORIS? OR CLASS OR CLASSES OR SUBJECTS OR AR-
             EAS OR TYPE)
S4    8786941   RULE? ? OR GUIDELINE? ? OR POLICY OR POLICIES OR PROCEDURE?
              ? OR DICTIONARY OR META()DATA OR META()DATA OR DATA(1N)(DESC-
             RIPTION OR TYPE? ?)
S5        148   S4(8N)ABSTRACTION
S6     104067   HIERARCH?
S7      98046   XML OR "C++"
S8          0   S2(10N)S5
S9      16723   S2(10N)S4
S10       691   S3(10N)S9
S11         0   S3(10N)S5
S12         0   S3(30N)S5
S13         0   S3 AND S5
S14         0   S10(10N)(STORAGE OR SERVER? OR DATA()WAREHOUSE OR DATABASE)
S15         0   (S6 OR S7)(10N)S10
S16         1   (S6 OR S7) AND S10
S17         1   S8 OR S13 OR S16
S18         1   RD  (unique items)
S19        49   S1 AND S5
S20        22   S19 NOT PY>2000
S21        21   RD  (unique items)
? t21/3,k/all

 21/3,K/1     (Item 1 from file: 9)
DIALOG(R)File   9:Business & Industry(R)
(c) 2006  The Gale Group. All rts. reserv.

01023106  Supplier Number: 23533146    (USE FORMAT 7 OR 9 FOR FULLTEXT)
NORWAY: Justice ministry looks into excess abstractions by power groups
(Okorim, which investigates ecological crime in Norway, wants list of power
  utilities which drew off more water from reservoirs than flow regulations
  allow)
Water Briefing, n 62, p 12
May 29, 1996
DOCUMENT TYPE: Newsletter  ISSN: 1352-6413  (United Kingdom)
LANGUAGE: English  RECORD TYPE: Fulltext
WORD COUNT:  230

TEXT:
...to NVE, Okokrim public prosecutor Jorn Holme says that his agency
```